SYSTEM AND METHOD FOR SIMULTANEOUS MEDIA PLAYOUT

Invented by
Sachin G. Deshpande

SYSTEM AND METHOD FOR SIMULTANEOUS MEDIA PLAYOUT

BACKGROUND OF THE INVENTION

1. Field of the Invention

This invention generally relates to multicast or unicast media communications and, more particularly, to a system and method for creating simultaneous media playout at multiple heterogeneous clients.

2. Description of the Related Art

In a streaming media system, a server typically streams media data to multiple clients. Each client has its own capability, which may be different than other clients. For example, each client may have different amount of buffering space available. The server streams the same media stream to all the clients in a multicast session. The goal is for all these clients with different buffering capabilities, is to achieve a simultaneous playback of the media stream. One application of a multicast session is a home-network environment where a single audio server streams a song to multiple client devices, perhaps in different rooms of a home, so that the clients achieve a simultaneous playout of a song.

Because each client has different buffering capabilities, the server needs to adapt its streaming whenever a new client joins the session or leaves the session. When a media packet is transmitted from the server to the clients, the packet may have a variable delay before reaching the clients. This variation in the end-to-end delay is called jitter. As noted by Schulzrinne et al., "RTP: A transport protocol for real-time applications", IETF, Jan. 2000, Sender and Receiver Report (SR, RR) packets of Real-time Transport Control Protocol (RTCP) part of Real-time Transport Protocol (RTP) have a field *interarrival jitter* that captures the mean deviation of the difference in packet spacing at the client compared to the server for a pair of packets. To compensate for the jitter, a streaming media client often buffers a certain number of packets, and then plays them out at scheduled playout times. A typical streaming media system often transports packets using RTP on top of User Datagram Protocol (UDP). UDP provides an unreliable service where transmitted packets can get lost, arrive out of order, or get duplicated. The client side buffering of data in a streaming media system helps to alleviate the out-of-order packet delivery by rearranging the buffered packets. The client can also discard the duplicate packets. The buffering is also helpful if there is any disruption in the streaming during the session, causing interruption in stream reception for a short time.

The amount of buffering done at the client side in a streaming media system is based mainly on two factors: the available client buffer size, and the acceptable delay which the user can tolerate before the media actually starts playing, after the time of the request. The type of media encoding used may also impose a certain amount of delay before a client can actually start to decode the received packets. This is typically the case for video encoded using any of the popular video encoding standards (e.g. MPEG1, MPEG2, MPEG4, H.263(+)), where the frames are encoded independently (Intra = I frames) or by referring to other frames (Inter = P,B frames). The delay may also be dependent upon whether the media is being played out is a *live* Stream or an *on-demand* archived stream.

With respect to the first factor, the available client buffer size, each client can have a different buffer size. In the case where each client is playing the same media stream in a session, it is assumed that packets are application data units and are independently decodable. This is true for majority of the audio encoding standards. The goal is to achieve a simultaneous playout of the media stream at each client. Another problem is the case where clients can join or leave the session midway. Client heterogeneity and dynamic session membership are major problems to be addressed in multicasting.

Adaptive playout delay adjustment and synchronization of streams have been proposed to address the above-mentioned problems. Assuming a media stream consisting of talkspurts interspersed with silence, adaptive playout algorithms have been proposed by Ramjee et al., "Adaptive playout mechanisms for packetizing audio applications in wide-area networks", Proceedings of IEEE INFOCOM, pp. 680-688, 1994. These algorithms estimate the mean and variation in end-to-end delay to adjust the starting time for playout of each talkspurt. However, this work does not address the issue of multiple heterogeneous clients having different buffering capabilities. It is also more targeted towards an interactive conferencing type of systems. Such a solution does not address an on-demand archived media (especially audio) distribution system, that has a continuous media stream without separate talkspurts. Neither does this method address server side adaptation based on different client buffering capabilities.

The other class of related work addresses the problem of synchronization between different media streams in a presentation session. For example, video and audio streams that need to be synchronously presented in a multimedia session. A start-up protocol to initiate synchronized playback of multiple media streams is proposed by Biersack et al., "Synchronized delivery and playout of distributed multimedia steams", Multimedia Systems, Vol. 7, No. 1, pp. 70-90, January 1999. A scheme that allows audio and video stream synchronization using a local conference bus, is also proposed by Kouvelas et al., "Lip Synchronization for use over the Internet", Proceedings of IEEE Globecom, Nov. 1996. These systems address a set of multiple streams transmitted from one or more servers to a single client, with the focus on achieving a synchronized playback of these multiple streams at that client.

Yuang et al., "Intelligent video smoother for multimedia communications", IEEE Journal of Selected Areas in Communications, Vol. 15, No. 2, pp. 136-146, Feb. 1997, describes an intelligent neural network based video smoother that compensates for jitter and smoothes the playout of video frames. However, these solutions do not address the issue of achieving simultaneous playback of the same media stream at multiple heterogeneous clients having different buffering capabilities.

Further, although prior art systems describe client side buffering, none of known solutions appear to handle simultaneous media playout at multiple clients, where each client has a different buffer capacity. Methods designed to achieve isochronous streams appearing simultaneously at output ports cannot handle simultaneous playout when the buffer capacity of each client is different.

It would be advantageous if media could be played out at several clients simultaneously, even if the clients had different buffering capacities.

It would be advantageous if simultaneous playout could be maintained despite disruptions in the media stream.

It would be advantageous if simultaneous playout could be maintained despite changes in client membership during a session.

SUMMARY OF THE INVENTION

The present invention avoids the above-mentioned problems associated with clients having different buffering capacities by permitting simultaneous media playout strategies for multiple heterogeneous clients using server side adaptation. There are three adaptation phase problems that are solved by the invention. The adaptive phases are: when a client joins the session; when a client leaves the session; and, after streaming disruptions.

Accordingly, a method has been provided for achieving simultaneous media playout in a network including a server and a plurality of clients. The method comprises: from a server, supplying a media stream to clients at a first bitrate (R1); determining the network delivery requirement; and, in response to the network delivery requirements, modifying the supply of the media stream.

Determining the network delivery requirements includes determining either the buffering capacities of the clients, or the media streaming disruptions. To determine the buffering capacities of the clients, a first minimum client buffering capacity (C1) is determined by polling the clients for their respective buffering capacities, and selecting the first minimum buffering capacity (C1) to be equal to the client with the smallest buffering capacity.

The method further comprises: following the supplying of media stream at a first bitrate (R1), changing clients in the network; determining the new minimum client buffering capacity (Cnew); and, in response to the new minimum buffering capacity (Cnew), modifying the supply of the media stream. The supply of the media stream is modified by temporarily pausing the supply of the media stream at the first bitrate (R1) if the new minimum buffering capacity (Cnew) is less than the first minimum buffering capacity (C1). Alternately, the media stream bitrate is temporarily increased if the new minimum buffering capacity (Cnew) is greater than the first minimum buffering capacity (C1), or if it has been determined that the supply of the media stream has been disrupted.

Additional details of the above-described simultaneous media playout method, and a system for achieving simultaneous media playout in a network are described below.

BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 is a schematic block diagram of the present invention system for achieving simultaneous media playout in a network.

Figs. 2 through 4 illustrate three examples, where a new client having a lower buffering capability joins the session midway (at time *m*), thus lowering the buffer capacity.

Figs. 5a through 5c illustrate additional examples of the present invention adaptation process.

Figs. 6 through 8 illustrate a scenario where a client leaves the session midway (at time *m*), thus increasing the buffering capacity.

Figs. 9a and 9b are flowcharts illustrating the present invention method for achieving simultaneous media playout in a network including a server and a plurality of clients.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

Fig. 1 is a schematic block diagram of the present invention system for achieving simultaneous media playout in a network. The system 100 comprises a server 102 having a network connection port to supply a media stream at a first bitrate (R1) on a network 104. The server 102 determines network delivery requirements and, in response, modifies the supply of the media stream. At least one client has a network connection port to receive the media stream. Shown are client one (106), client two (108), and client *M* (110). However, the invention is not limited to any particular number of clients. The clients 106-110 transmit their buffering capacities to the server 102.

The server 102 determines the network delivery requirements in response to determining the buffering capacities of the clients. Further, the server determines network delivery requirements in response to determining media streaming disruptions.

With respect to determining the buffering capacity of the clients 106-110, the server 102 determines the first minimum client buffering capacity (C1). The server 102 polls the clients 106-110 for their respective buffering capacities and each client 106-110 transmits their buffering capacity to the server 102, in response to the poll. The server 102 determines the first minimum buffering capacity (C1) to be equal to the client with the smallest buffering capacity.

Each client schedules the playout to be at a time equal to a first time interval (t1) plus the minimum buffering capacity divided by the first bitrate (C1/R1). Prior to supplying a media stream at the first bitrate (R1), the server 102 communicates the first minimum buffering capacity (C1) to the clients. Each client receives the value t1 as a timestamp of the first buffered frame in the media stream. As part of the normal playout procedure, each client 106-110 reorders out-of-order media packets, handles any lost media packets, and schedules the playout of the media packets at a uniform rate in response to the media type and the first bitrate (R1) because the media stream includes media packets with a timestamp.

One major problem addressed by the present invention is that of adding, or removing clients from the network. For example, after supplying the media stream at a first bitrate R1 to client one (106) and client two (108), an additional client, client *M* (110) is added to the network 104. The server 102 determines the new minimum client buffering capacity (Cnew), in response to the addition of the new client 110 and communicates the new minimum buffering capacity (Cnew) to the clients 106-110. Then, the server 102 modifies the supply of media stream in response to the new minimum buffering capacity (Cnew).

The addition or removal of clients from the network can either raise or lower the new minimum buffering capacity. If the buffering capacity is lowered, the server 102 modifies the supply of the media stream by temporarily pausing the supply of the media stream at the first bitrate (R1). If the buffering capacity is increased, the server 102 temporarily increases the media stream bitrate. Prior to modifying the supply of the media stream, the server 102 communicates the new minimum buffering capacity (Cnew) to the clients 106-110.

More specifically, the server 102 temporarily pauses the supply of the media stream if the new minimum buffering capacity (Cnew) is less than the first minimum buffering capacity (C1). The server 102 temporarily pauses the supply of the media stream by pausing for a time equal to the first minimum buffering capacity minus the new minimum buffering capacity, divided by the first bitrate ((C1 – Cnew)/R1). Typically, the server 102 communicates media stream modifications to the clients using real-time-streaming protocol (RTSP). However, the present invention is not limited to any particular type of protocol. There are many pre-exiting protocols that can be modified for use with the system 100.

The server 102 temporarily increases the media stream bitrate if either the new minimum buffering capacity (Cnew) is greater than the first minimum buffering capacity (C1), or the server 102 determines that the supply of the media stream has been disrupted. The server 102 temporarily increases the media stream bitrate if the new minimum buffering capacity is greater than the first minimum buffering capacity by polling the clients and itself to determine the minimum bitrate (Rmin). That is, the server 102 determines the maximum rate at which each client can receive a media stream, and the rate at which it can transmit. The device with the smallest bandwidth is the gating device, and the minimum bitrate (Rmin) is set to be equal to the smallest bandwidth. Note, that the minimum bitrate (Rmin) is greater than the first bitrate (R1). Then, the server 102 supplies the media stream at the minimum bitrate (Rmin) for a time equal to the new minimum buffering capacity minus the first minimum buffering capacity, divided by the minimum bitrate minus the first bitrate ((Cnew – C1)/(Rmin – R1).

To ease the above-mentioned calculations, the present invention uses the concept of a current_buffer_level and a target_buffer_level. The target_buffer_level is the same as the first minimum buffering capacity. Typically, the server 102 supplies media stream data as media packets. The server 102 first determines the current buffer level (Cc), which is the number of media packets buffered at each client 106-110. If the new minimum buffering capacity is less than the current buffer level, the server temporarily pauses the supply of media stream for a time equal to the current buffer level minus the new minimum buffering capacity, divided by the first bitrate ((Cc – Cnew)/R1). This calculation leads to the same results as the equation ((C1 – Cnew)/R1) mentioned above.

If the new minimum buffering capacity is greater than the current buffer level, the server 102 polls the clients 106-110 and itself to determine the minimum bitrate (Rmin), as mentioned above, and supplies the media stream at the minimum bitrate (Rmin) for a time equal to the new minimum buffering capacity minus the current buffer level, divided by the minimum bitrate minus the first bitrate ((Cnew – Cc)/(Rmin – R1). The equation leads to the same results as equation ((Cnew – C1)/(Rmin – R1) mentioned above.

The server 102 also temporarily increases the media stream bitrate if it has been determined that the media stream has been disrupted. The server 102 polls the clients 106-110 and itself to determine the minimum bitrate (Rmin), and supplies the media stream at the minimum bitrate (Rmin) for a time equal to new minimum buffering capacity minus the current buffering capacity, divided by the minimum bitrate minus the first bitrate ((Cnew – Cc)/(Rmin – R1).

In some aspects of the invention, the server 102 includes a tracker 110 to maintain a current-buffer-level measurement of the number of media packets being supplied by the server and a target-buffer-level measurement of the new minimum client buffer. The server 102 modifies the supply of the media stream in response to the current-buffer-level and target-buffer-level measurements maintained by the tracker 110, using the equations mentioned above.

Below, some examples are given of the above-mentioned system 100 in operation, and additional details to understand the problems solved with the use of the present invention.

Adaptation phase when a client joins the session on the network

Assume that the media is encoded, although the same analysis holds true for uncompressed media data, at a bitrate of R1 bits per second (bps). Frame-based media encoding is also assumed. Various audio coding standards (e.g. G.723, G.728, G.729, GSM, MP3, etc.) fall under this category. Also assume that there are M heterogeneous clients having buffer sizes $\{C_1, C_2, ..., C_M\}$ bits, currently receiving the media stream. Every time a new client joins to receive the current media stream, the server queries its buffering capabilities. Assuming the client and server to be using Real-time Streaming Protocol (RTSP), the server could query the client buffering capability using a transaction similar to one shown below.

Getting Client Buffer Size:

Server $\rightarrow$ Client :

```
GET_PARAMETER rtsp://myhomenw.com/filez/song1.mp3
RTSP/1.0
CSeq: 1089
Content-type: text/parameters
Session: 8438
Content-length: 18
CRLF
buffer_size_bits
```

```
CRLF
```

Client$\rightarrow$Server:

```
RTSP/1.0 200 OK
CSeq: 1089
Content-length: 24
Content-type: text/parameters
CRLF
buffer_size_bits: 64000
CRLF
```

The server keeps track of the buffering capabilities of each client. Let C1 bits denote the minimum buffer size of the currently participating clients, i.e. C1 $= Min(C_1, C_2, ..., C_M)$. It is assumed that each media packet streamed has a sequence number and a timestamp field (each RTP packet has a sequence number and a timestamp field). The sequence number can be used at the client side to detect duplicate, out-of-order and missing packets. The timestamp can be used to keep track of packet jitter and also to determine the playout scheduling of packets. Every time the buffering capacity changes as a result of a new client joining the media stream, with the buffering capability being equal to the least of the currently existing clients, the server communicates the new minimum buffering capacity (Cnew) value to all the clients. Assuming the server and client to be using RTSP, this can be done using a transaction similar to one shown below.

Setting Client's Buffering Level:

Server$\rightarrow$Client:

```
SET_PARAMETER rtsp://myhomenw.com/filez/song1.mp3 RTSP/1.0
CSeq: 1094
Content-length: 18
Content-type: text/parameters
CRLF
buffering_size: 38400
```

```
CRLF
```

Client $\rightarrow$ Server :

```
RTSP/1.0 200 OK
CSeq: 1094
Content-length: 7
Content-type: text/parameters
CRLF
38400
CRLF
```

It is assumed that the server has also communicated the media stream first bitrate R1. This will typically be done as a part of the Setup process, or could be done using the SET_PARAMETER method as above (assuming the server and clients to be using RTSP). The media stream itself may already carry the bitrate information (e.g. MP1, MP2, MP3, G.723.1).

Every time a new client joins the session, after going through the setup and capabilities exchange process, the new client :

- Buffers and re-orders the incoming packets,
- Schedules the playout of the first buffered frame with timestamp = $t1$, at time $(t1+\frac{C1}{R1})$,
- Schedules the playout of the next buffered frames in order at the uniform rate based on the type of media encoding and the first bitrate R1.

Every time a new client joins, the server goes through the steps of setup and capabilities exchange. If the new client changes (lowers) the buffering capacity, assuming C1 and Cnew refer to the old and the new minimum client buffer space, the server:

- Communicates the Cnew value to all the clients.

- Pauses the media transmission for time $(\frac{(C1-Cnew)}{R1})$ and then restarts the media delivery at the usual rate of $R1$. If the server and client are using RTSP, then this pause in the media transmission is equivalent to a request from client as shown below :

```
Pause rtsp://myhomenw.com/filez/song1.mp3 RTSP/1.0
CSeq: 1145
Range: npt=48
Session: 8438
```

where in the above example, the pause occurs at time 48 seconds and will last for $(\frac{(C1-Cnew)}{R1})$ seconds.

During the adaptation period (pause period) of the server:

- If a new client joins, thus lowering the current buffering capacity, the server can just augment the pause (increase it) based on the value of Cnew and communicate the Cnew value to the clients. Denoting the old Cnew value as C1, and the newer value as Cnew ((Cnew)newer). The total pause time duration, including the already paused time, will be $(\frac{C1-Cnew}{R1})$.
- If a client leaves, thus increasing the current buffering capacity, the server will complete the calculated pause before reacting to this buffering capacity change. The adaptation at the end of the pause period will be done according to the procedure described in the next section.

Figs. 2 through 4 illustrate three examples, where a new client having a lower buffering capability joins the session midway (at time *m*), thus lowering the buffer capacity. The first example (Fig. 2) shows packet transmission at the server, packet reception at the client, and the packet playout schedule at the client in the case when there is no interarrival packet jitter, no out-of-order packet delivery, and no packet loss. The second example (Fig. 3) illustrates the situation of Fig. 2 with packet jitter and out-of-order packet delivery. The third example (Fig. 4) includes packet jitter, out-of-order packet delivery and packet loss. As can be seen from the figures, buffering can absorb the packet jitter and out of order packets. A packet loss could be handled using some receiver-based method. This can be a simple repetition of the last packet, a silent frame insertion, or some complex interpolation of data surrounding the missing data.

Adaptation phase when a client leaves the session

Every time a client leaves the session (TEARDOWN message received, if using RTSP), the server computes new value of the minimum buffering capacity Cnew. If this value is higher than the previous value (C1), the server computes minimum of the maximum available reception bandwidth amongst all the clients. This is achieved by querying the maximum client bandwidth (bitrate) using GET_PARAMETER method. Based on the type of connection, this bandwidth (bitrate) may vary over time, so the server uses the most recent estimate of the client bandwidth (bitrate). Assuming the server and client using RTSP, this estimate can also be obtained from the client in its Bandwidth header field {section 12.6 of the real time streaming protocol (RTSP)}. Let $Rclient = Min\ (Rclient1, Rclient2, ..., RclientM)$ be the minimum of the maximum available client bandwidths and assume that $R\min > R1$. Let R_{server} be the maximum rate at which the server can stream media data. It is assumed further that the media stream being transmitted is already encoded and is available completely (as in case of an on-demand archived stream). Then the server can stream at a higher rate of $R\min = Min(Rclient, R_{server})$ bps, for time $T_h = \frac{Cnew - C1}{R\min - R1}$ seconds and then revert back to the transmission rate of R1 bps. If the server and client are using RTSP, then this change in speed of transmission is equivalent to the requests from the client as shown below :

```
Play rtsp://myhomenw.com/filez/song1.mp3 RTSP/1.0
CSeq: 1150
Range: npt=104-120
Speed: 1.2
Session: 8438

Play rtsp://myhomenw.com/filez/song1.mp3 RTSP/1.0
CSeq: 1150
Range: npt=120-
Speed: 1.0
Session: 8438
```

Assuming that $\frac{R\min}{R1} = 1.2$, and $\frac{Cnew - C1}{R\min - R1}$ = 16 seconds, the server transmission change occurs at time 104 seconds. The server also communicates Cnew to all the clients (which being higher than previous Cnew (now called C1) can be used by the clients to get ready to receive media data at a higher rate).

Figs. 5a through 5c illustrate additional examples of the present invention adaptation process. During the adaptation period (increased rate transmission time period) of the server, if a newer client joins at time t, thus lowering the buffering capacity:

- Case 1: If $(Cnew < (R\min - R1)t + C1)$ (Fig. 5a and 5b), the server immediately pauses the transmission for time $(\frac{(C1 + (R\min - R1)t - Cnew)}{R1})$, where t is the total duration the media was transmitted at rate $R\min$. After this pause the server reverts back to the original transmission rate of $R1$.

- Case 2: If $(Cnew > (R\min - R1)t + C1)$ (Fig. 5c), the server calculates a newer value of Rmin, and streams at this increased rate for time $\frac{(Cnew - C1) - (R\min - R1)t}{(R\min - R1}$. After this increased rate transmission time period the server reverts back to the original transmission rate of $R1$.
- Case 3: If $(Cnew = (R\min - R1)t + C1)$, server reverts the transmission to the rate $R1$.

The implementation of the above three cases can be simplified by assuming a hypothetical buffer at the server side. Two variables: current_buffer_level and target_buffer_level can be used to keep track of the fill level and the target level of the hypothetical buffer. If a newer client joins, thus lowering the buffering capacity, the target_buffer_level is set to the Cnew. Then, the three cases above can be implemented as follows.

- Case 1: If (target_buffer_level < current_buffer_level), the transmission pauses for time $\frac{(\texttt{current\_buffer\_level} - \texttt{target\_buffer\_level})}{R1}$, then reverts back to the transmission at the rate of $R1$.
- Case 2: If (target_buffer_level > current_buffer_level), a new Rmin is found, where $R\min = Min(Rclient, R_{server})$, and the media stream is supplied at this increased rate for time $\frac{(\texttt{target\_buffer\_level} - \texttt{current\_buffer\_level})}{R\min - R1}$. Then, the media stream supply reverts back to rate $R1$.

- Case 3: If (target_buffer_level = current_buffer_level), the media stream supply reverts back to the rate $R1$.
- If a client leaves, thus increasing the buffering capacity, the server does not react to this change until the end of the current increased speed interval. The server acts on this new change (with another possible increased speed transmission interval) at the end of the current interval.

When a client leaves the session, thus increasing the buffering capacity, the server may choose to do no adaptation (increased rate transmission). This maybe because some clients do not support increased bandwidth reception, or because the server cannot handle the increased speed transmission adaptation phase for some reason. In this case, the system will work with a sub-optimal buffering behavior, which may result in some missed playout schedules. Essentially the system will continue to behave as if the lower buffering capacity client is still active.

Figs. 6 through 8 illustrate a scenario where a client leaves the session midway (at time *m*), thus increasing the buffering capacity. The first case (Fig. 6) shows packet transmission at the server, packet reception at the client, and the packet playout schedule at the client in the case when there is no interarrival packet jitter, no out-of-order packet delivery, and no packet loss. The second case (Fig. 7) shows the same in the presence of the packet jitter and out-of-order packet delivery. The third case (Fig. 8) includes packet jitter, out-of-order packet delivery, and packet loss.

Adaptation phase after streaming disruptions

During the transmission session, the network load or competing traffic can result in congestion, and the session throughput from the server to clients can fall for a short duration. In this case after the congestion is over, the server can go through an adaptation phase similar to the adaptation phase when a client leaves the session, as mentioned above. As before, the period of high rate transmission can be calculated through the two variables: current_buffer_level and target_buffer_level, to keep track of the fill level and the target level of the hypothetical buffer at the server side. The target_buffer_level is equal to the minimum client buffer size and the current_buffer_level is determined by keeping track of the data streamed from the server and played out at the clients. In this case, because of the stream disruption, (target_buffer_level > current_buffer_level), the server can find a new Rmin, where $R\min = Min(Rclient, R_{server})$, and stream at this increased rate for time $\frac{(\texttt{target\_buffer\_level} - \texttt{current\_buffer\_level})}{R\min - R1}$. The server then reverts back to rate $R1$. If a new client joins during this adaptation phase, thus lowering the buffering capacity, target_buffer_level is set to the newer minimum buffering capacity Cnew, and the 3 cases as described in the previous section are checked and acted on accordingly. This adaptation phase can also be carried out every time the difference (target_buffer_level - current_buffer_level) exceeds some threshold.

Figs. 9a and 9b are flowcharts illustrating the present invention method for achieving simultaneous media playout in a network including a server and a plurality of clients. Although the method is depicted as a sequence of numbered steps for clarity, no order should be inferred from the numbering unless explicitly stated. The method begins at Step 900. Step 902, from a server, supplies a media stream to clients at a first bitrate (R1). Step 904 determines the network delivery requirement. Step 906, in response to the network delivery requirements, modifies the supply of the media stream.

Determining the network delivery requirements in Step 904 includes determining the buffering capacities of the clients, or determining media streaming disruptions. When determining the buffering capacities of the clients, the first minimum client buffering capacity (C1) is determined. Determining the first minimum client buffering capacity (C1) in Step 904 includes substeps. Step 904a polls the clients for their respective buffering capacities. Step 904b determines which client has the smallest buffering capacity. Step 904c selects the first minimum buffering capacity (C1) to be equal to the client with the smallest buffering capacity.

A further step, Step 908, at each client, schedules the playout to be at a time equal to a first time interval (t1) plus the minimum buffering capacity divided by the first bitrate (C1/R1). Step 901, prior to supplying a media stream at the first bitrate (R1), communicates the first minimum buffering capacity (C1) to the clients.

Following the supplying of media stream at a first bitrate (R1) in Step 902, Step 905a changes clients in the network. Step

905b determines the new minimum client buffering capacity (Cnew). Step 905c communicates the new minimum buffering capacity (Cnew) to the clients. Step 906, in response to the new minimum buffering capacity (Cnew), modifies the supply of the media stream. Modifying the supply of the media stream in Step 906 includes temporarily pausing the supply of the media stream at the first bitrate (R1), or temporarily increasing the media stream bitrate.

Temporarily pausing the supply of the media stream at the first bitrate (R1) in Step 906 includes temporarily pausing the supply of the media stream if the new minimum buffering capacity (Cnew) is less than the first minimum buffering capacity (C1). More specifically, Step 906 includes pausing for a time equal to the first minimum buffering capacity minus the new minimum buffering capacity, divided by the first bitrate ((C1 – Cnew)/R1).

Determining the new minimum buffering capacity (Cnew) in Step 905b includes substeps. Step 905b1 determines if the new minimum buffering capacity (Cnew) is greater than the first minimum buffering capacity (C1). Step 905b2 determines if the supply of the media stream has been disrupted. Modifying the supply of the media stream in Step 906 then includes temporarily increasing the media stream bitrate.

Step 905d polls the clients and the server to determine the minimum bitrate (Rmin). Then, modifying the supply of the media stream in Step 906 includes supplying the media stream at the minimum bitrate (Rmin) for a time equal to the new minimum buffering capacity minus the first minimum buffering capacity, divided by the minimum bitrate minus the first bitrate ((Cnew – C1)/(Rmin – R1).

In some aspects of the invention, Step 905e, at the server, maintains a current_buffer_level measurement to track the number of media packets supplied by the server. Step 905f, at the server, maintains a target_buffer_level measurement to track the new minimum client buffering capacity (Cnew). Modifying the supply of the media stream in Step 906 includes modifying the supply of the media stream in response to the current_buffer_level and target_buffer_level measurements.

Supplying a media stream from a server to clients at a first bitrate (R1) in Step 902 includes supplying data as media packets with a timestamp. Step 905g determines the current buffer level (Cc), which is the number of media packets buffered at each client. Step 905h determines if the new minimum buffering capacity is less than the current buffer level. Modifying the supply of the media stream in Step 906 includes temporarily pausing the supply of media stream for a time equal to the current buffer level minus the new minimum buffering capacity, divided by the first bitrate ((Cc – Cnew)/R1).

Step 905d still polls the clients and the server to determine the minimum bitrate (Rmin). Modifying the supply of the media stream in Step 906 includes supplying the media stream at the minimum bitrate (Rmin) for a time equal to the new minimum buffering capacity minus the current buffer level, divided by the minimum bitrate minus the first bitrate ((Cnew – Cc)/(Rmin – R1).

Determining the current_buffer_level in Step 905g includes determining that the media stream has been disrupted. Then, modifying the supply of the media stream in Step 906 includes supplying the media stream at the minimum bitrate (Rmin) for a time equal to new minimum buffering capacity minus the current_buffer_level, divided by the minimum bitrate minus the first bitrate ((Cnew – Cc)/(Rmin – R1).

Communicating the media stream modifications to the clients from the server in Steps (and Substeps of) 904 and 905 typically includes using real-time-streaming protocol (RTSP).

Scheduling the playout to be at a time equal to a first time interval (t1) plus the minimum buffering capacity divided by the first bitrate (C1/R1) in Step 908 includes substeps. Step 908a, at each client, reorders out-of-order media packets. Step 908b, at each client, handles any lost media packets. Step 908c, at each client, schedules the playout of the media packets at a uniform rate in response to the media type and the first bitrate (R1).

A system and method have been provided for adapting the distribution of media for simultaneous playout with heterogeneous clients. Some examples have been given of controlling the supply of media stream through the tracking of client buffering capacities. However, other variations and embodiments of the present invention will occur to those skilled in the art.

WE CLAIM: